■ ASPLOS **四日目**

本会議二日目.
今回は時差をうまくのりこえたと思ったのだけど,
夕方ものすごい睡魔におそわれてしまって会議終了後にダウン.
バンケットにいきそびれてしまった ...

■ Keynote: Multicore Programming

- multicore issues
  - good performance when synchronization is required (eg. database)
- multicore machine
  - sort of distributed <-> cores share memory, and do not fail independently
- MassTree
  - high-performance key/value store
    - eurosys 12 "cache craftiness for fast multicore key-value storage"
    - https://pdos.csail.mit.edu/papers/masstree:eurosys12.pdf
  - concurrency control
    - write(locking) -> compare and swap, on one memory word, v#
    - read(no locking)
- Silo
  - high-performance database, with transactions
    - speedy transactions in multicore in-memory databases sosp13 - http://db.csail.mit.edu/pubs/silo.pdf
    - fast durability and recovery through multicore parallelism osid14 - https://www.usenix.org/system/files/conference/osdi14/osdi14-paper-zheng_wenting.pdf
  - database is in primary memory, runs one-shot requests
  - optimistic concurrency control, thread maintains read-set and write-set -> at end, attempts commit
- STO (software transactional objects)
  - type-aware transactions for faster concurrent code, eurosys 16 - https://tslilyai.github.io/sto.pdf
  - a vision for concurrent code
    - apps run transactions
    - using transaction-aware datatypes - sets, maps, arrays, boes, queues
    - moving transactional memory up a level
  - performance opportunities
    - transaction-aware types provide: smaller read-and write-sets, relaxed checks, installs that do computation
- Observations
  - this work depended on abstract data types - better performance by taking advantage of semantics
  - concurrency control should be the job of the implementation
  - encapsulation is crucial

■ Persistent Memory

PMTest: A Fast and Flexible Testing Framework for Persistent Memory Programs

- support for crash consistency have tow fundamental guarantees
  - durability: writes become persistent in PM
  - ordering: one write becomes persistent in PM before another
- flexible
  - prior works only support specific software and hardware
  - <- durability と ordering だけみればいい , write, clwb, sfence

- fast
  - piro works uses exhaustive testing
  - <- PMTest infers the persistens interval from PM operation trace
- interface
  - expoert: assertion-like low-level interface
    - isOrderBefore, isPersisted
  - normal: high-level interface, automatically iject low-level chckers
    - checkers for PMDK transaction, TX_CHECKER_START - TX_CHECKER_END
    - - crash consistency bugs, performance bugs

Finding and Fixing Performance Pathologies in Persistent Memory Software Stacks

- paper は battecy-backuped な DIMM だけど発表は Optane
- PM-aware file system - bpfs, pmfs, nova, strata
  - legacy codes built for disk run slow ihon PM
  - what are the best ways to optimize software system on PM?
- fix urgent problems, provide est practices for optimization
- contribution(1): analyze a ragen of optimization tech
- FLEX: File emulation with DAX - emulate POSIX IO in userspace
  - RocksDB, SQLite - use file to implement WAL for consistency
    - 変更は RocksDB = 56LOC, SQLite = 233LOC
- PM data structure
  - ralated with NovelLSM, SLM-DB, Level hashing, CCEH, NV-Tree, FP-tree
- contribution(2): why do wee need another new file system?
  - key overhead: block-based legacy journaling device
  - -> jounaling DAX Device(JDD)
- contribution(3): improve scalability for PM file system
  - mmeory-entric optimizations - NUMA-aware file access in NOVA

Fine-Grain Checkpointing with In Cache Line Logging

- design a durable data structure for NVM <- cache can reorder writes
  - existing - explicitly force a write back (fflush) -> expensive
  - <- periodic persistency, in cache line log(InCLL)
    - zero explicit writes back
- periodic persistency
  - flush entier cache infrequentrly(e.g., every 64ms) - x86's wbinvd
  - return to a consistent state at the end of an apoch - using undo log
- In Cache Line Log
  - benefits: a cache line is evicted to meory atomically, no explicit write back necessary
    - enables recovery without explicit write back
  - drawback: capacity is very limited
- External Undo Log + In Cache Line Log
- cf. https://github.com/epfl-vlsc/Incll

■ Accelerators

FA3C: FPGA-Accelerated Deep Reinforcement Learning

- inference & training, 32-bit precision float/no weight pruning, embedded / datacenter (better perf. than GPU)
- ref. A3C ICML2016 - http://proceedings.mlr.press/v48/mniha16.pdf

- small computation batch size; inference = 1 / training = 5, limited off-chip B/W; n versions of local parameters, kernel launch overhead: frequent kernel launches
- -> tailored datapath for small-batch size
- MAC にうまくデータ供給できるようなラインバッファを構築
- VCU1525 で P100 より performance/energy でよい.

AcMC^2: Accelerating Markov Chain Monte Carlo Algorithms for Probabilistic Modeling

- accelerable kernels: random number gennerators
  - expert optmized FPGA URNG, 4bit LFSR, 1cycle latency, 1op/cycle
  - traditional RNGs - cryptographically secure, rejection sampleing: stalls
- identifying parallelism: enter markov blankets
- identifying parallelism: k-Colorings

AcMC^2 は Probabilistic models をハードウェアにコンパイルする．エキスパートの作ったテンプレートを組み合せる, 並列性抽出する, など． gitlab.engr.Illinois.edu/DEPEND/AcMC2 で 公開？？

Targeting Classical Code to a Quantum Annealer

- ゲートを D-Wave にマッピング
- Veirlog → EDIF → D-Wave で実行
- https://github.com/lanl/edif2qmasm
  - cf. edif2qmasm makes it possible to run Verilog or VHDL programs on a D-Wave quantum annealer.

■ Graph Processing

PnP: Pruning and Prediction for Point-To-Point Iterative Graph Analytics

Point-to-Point Query に動的な枝刈りと予測を導入． Quegel(VLDB '16) より高速に.

DiGraph: An Efficient Path-based Iterative Directed Graph Processing System on Multiple GPUs

複数 GPU で効率的に有向グラフを処理できるように path-based asynchronouse execution を導入.

Phoenix: A Substrate for Resilient Distributed Graph Analytics

- Fail-stop faults からの復帰を考慮したグラフアルゴリズム.
- 分散グラフアルゴリズムを分類
  - Self-stabilizing graph algorithms
  - Locally-correcting graph algorithms
  - Globally-correcting graph algorithms
  - Globally-consistent graph algorithms

■ Microarchitecture

Characterizing Latency, Throughput, and Port Usage of Instructions on Intel Microarchitectures

- peformance mystery - Intel のコアの命令の組合せでの性能が謎
- port usage を観測できるようなマイクロベンチマークを生成
- Nehalem から Cannon Lake までで実行
- http://uops.info/
- https://github.com/andreas-abel/nanoBench

Bootstrapping: Using SMT Hardware to Improve Single-Thread Performance

- decoupled lookahead arechitecture(DLA)
- naive implementation of DLA on SMT ineffective
  - resource contention make naive approach ineffective
- -> dynamically allocate on-chip resources

CORF: Coalescing Operand Register File for GPUs

1, 2, 3Byte しか使ってないレジスタをうまくパックして省電力化につなげる話

■ WACI

- VR Swarms，面白そう
- Genetic Programming，ちょっとさわっておかないと，かな.
- Unfare なデータセンタ ... それってどうなんだろうなあ，とか.